Exhibit 17

Filed Under Seal

Message

From: Sherwin Liu [Sherwin.Liu@sonos.com]
Sent: 7/7/2014 6:54:14 AM
To: Tad Coburn [Tad.Coburn@sonos.com]; Debajit Ghosh [debajit@google.com]
CC: Umesh Patil [upatil@google.com]; Christopher Patnoe (patnoe@google.com) [patnoe@google.com]; Adam Graham [adam.graham@sonos.com]
Subject: RE: CloudQueue API

Hi Debajit,

Hope all goes well with you and your team!

I was hoping to check on the status of the CloudQueue API. What's the expected timeframe for publishing this API? We are about to start building a prototype, so the closer it is to the actual API the better.

Also, just another reminder that I'll be covering for Tad in the next two weeks, so please include me in any CloudQueue discussions. :)

Thanks,

-Sherwin

-----Original Message-----
From: Tad Coburn
Sent: Wednesday, July 02, 2014 5:05 PM
To: Debajit Ghosh; Sherwin Liu
Cc: Umesh Patil; Christopher Patnoe (patnoe@google.com); Adam Graham
Subject: RE: CloudQueue API

Debajit -

Sorry for the slow response on this thread - this is a key one for us at Sonos, since we're about to start working on our player implementation for cloud queue playback using the information you've provided so far via email.

This email has three sections:
- a note about my upcoming time off and who will cover for me
- responses to your previous comments
- a few new questions re: the cloudqueue data model and API

==Tad's upcoming time off==
I'll be out of the office starting next Tues, July 8, through Sunday, July 20th. While I'm out, Sherwin Liu will be your technical contact for all things related to cloud queue design and implementation. I will check email the first week (up thru 7/11), but will be totally offline the second week.

==Response to your previous comments==
"repeat" and "shuffle" modes:
I can understand why you think of shuffle and repeat as going hand-in-hand and thus should both be implemented on the server. After thinking about it some more, I don't have an objection to that; however, the Sonos player does need a way to:
a) determine the current playback "mode" (shuffle=on/off/disabled, repeat=on/off/disabled)
b) determine when the current playback mode changes (perhaps as part of the polling operation?) and ideally a way to:
c) set each of the modes (shuffle, repeat) to a new value (on or off) (IFF that mode is not disabled).

==New Questions==
1) Will the cloud queue "keep track" of the currently playing track? If so, I assume the player will call another REST API each time a new track starts playing. Any thoughts on how that would look in a REST API? Would it be a PUT to http://.../cloudqueue/currenttrackiid or something like that?
Will the player also need to report any "seek" operations within the track (i.e. "user just jumped to time offset 1 minute 27 seconds within the track")? That might warrant a slightly more general resource http://.../cloudqueue/currenttrackposition to which you could PUT both a track IID and a time offset within that track.

2) How do you see the "jump to a new track" operation working? This is the scenario where the user views the list of tracks in the queue, scrolls down and touches a new track, which should immediately cause that new track to begin playing. I can think of two options:

2a) GPM app tells the CloudQueue that there is a new "targetTrackIID" value, and then tells the Sonos player to immediately check for changes to the cloudqueue. The latter would use the same out-of-band notification that is also used when the currently playing track is deleted. The cloudqueue would store the new targetTrackIID value, and when the client asks for a window of tracks, the cloudqueue service would ignore the currentTrackIID argument and instead use the targetTrackIID value it had stored, and return a window of tracks containing that. The cloudqueue would also need to return the targetTrackIID value so the Sonos player would know to start playing that track immediately.
2b) GPM app tells the Sonos player to jump to track <tiid> using some new TBD API. The Sonos player then immediately fetches an updated window of tracks and resumes playing starting with track <tiid>.

3) Is there any other state that the player will need to sync with the cloud queue? So far you have:
  <list of tracks and their metadata>
  <revision token for list of tracks>
  <shuffle on/off/disabled>
  <repeat on/off/disabled>
  ? <currentTrackIID> (if the cloudqueue keeps track of the currently playing track)
  ? <targetTrackIID> (used when jumping to a new track, if you use approach (2a) above) Do you plan to keep the PLAYING/PAUSED state as an attribute on the cloudqueue? If yes, will the Sonos player be responsible for updating that state and/or for detecting when that state has changed?

Thanks!

- Tad

-----Original Message-----
From: Debajit Ghosh [mailto:debajit@google.com]
Sent: Wednesday, June 25, 2014 2:06 AM
To: Tad Coburn
Cc: Umesh Patil; Christopher Patnoe (patnoe@google.com); Adam Graham
Subject: Re: CloudQueue API

thanks! responses inline:

On Mon, Jun 23, 2014 at 2:25 PM, Tad Coburn <Tad.Coburn@sonos.com> wrote:
> Debajit -
>
> Thanks as always for the quick reply!
>
> - understood about the REST URLs. That makes sense.
>
> - version id is a token: should be no problem as long as it has a
> finite length, plus ideally isn't too large. <= 128 bytes would be
> ideal for us :-)
>

sg. i'll ask the team what their thoughts are on the contents for this particular token.

> - we can look for an "Authorization: <token>" header in the custom headers if necessary. If we do that, then I'm assuming your service will return a new token to our client via a response header something like "Google-Updated-Authorization: <token>" and we'll use "<token>" as the new value for the "Authorization:" header we send in subsequent requests.

correct.

>
> - working out the details of the commands in parallel sounds reasonable. I agree we can wait a little bit and work out more of the new SSDP/WebSocket-based protocol before deciding whether to start with an MRP-based integration or jump straight to the new WebSocket protocol integration.
>
>
> New questions:
> - please remind me: can the OAuth token get refreshed when Sonos goes to fetch one of the track URIs, or will it only be refreshed when Sonos calls one of the cloud queue REST methods? It's not a super-big deal for us either way, but would be helpful to know for planning purposes.

the plan is to allow for refresh on any communication with Google servers (cloud queue server or stream authentication request). we'll have to check that this is possible, but we don't see any issues here yet.

>
> - are you hoping to implement a race-free "play next" operation (which would require some tight cooperation with the Sonos player/receiver) or are you going to use a heuristic/best-effort approach to try to ensure that the new tracks get inserted at the right position? I'm trying to figure out how much

work we need to plan for that feature. It can be done atomically only if you pass the list of inserted tracks (or a window-sized subset thereof) to the player/receiver first so that it can atomically insert them in the correct place in its local cache and then tell your app the ID of the track after which they were inserted. Any other approach runs the risk of inserting them after what the cloud queue thinks is the current track, but the player has already moved on to the next track and thus the newly inserted tracks are skipped over.

good use case -- this is one of the tricky cases we were planning on handling as best effort. we were thinking of having the sender use its knowledge of what's currently playing (we could have it get the current status explicitly, if desired) and otherwise be responsible for issuing the queue mutation request to the cloud queue. the receiver would then find out about the change in its next poll. does that sound reasonable?

>
> - shuffle and repeat mode (lower priority right now, but we should start thinking about them):
> -- I believe repeat should be implemented by the Sonos player (a.k.a. receiver) since it is simple to implement and it avoids making the cloud queue appear to be an "infinite" list of tracks in cases where it is really a finite playlist.

interesting - was definitely thinking of having both shuffle and repeat be handled by the cloud queue, so it could return the appropriate content on subsequent calls to get the next window of tracks. doing so is a bit more complicated, but it allows the receiver and the clients not to have to implement the same logic (or worry about using the same random seed, etc., in the case of shuffle).
e.g., we'll want to make sure that the queue shown in the sender's ui, and the actual tracks playing on the receiver are consistent.

and, for whatever reason, i can't easily separate shuffle and repeat in my mind (it seems like if you solve shuffle, you solve repeat -- especially when you're handling both repeat and shuffle).

btw, we disable shuffle/repeat in our ui when the user is playing a radio station -- another reason this might best be encapsulated on the server-side.

what do you think?

> -- I think we already agreed that shuffle should always be implemented by the cloud queue, and that the player/receiver should always play the tracks in the linear order that the cloud queue presents to it.

agreed

> -- I think we should plan on making it possible for both the GPM app and the Sonos app to display and change the shuffle/repeat mode/state. Do you agree? I'd be ok if the player/receiver only discovers a change in this state when it polls for the cloud queue revision number, although a low-latency event from the GPM app to the player would be ideal from a user perspective.

agreed in general (hadn't thought that this would go into the controller ui), tho i guess that means the receiver needs to be able to get the shuffle/repeat state from the cloud queue (the sender will need to be able to set/get that). and as per above, the cloud queue would need to be able to tell the receiver that none of those modes should be displayed, period, if the user is playing a radio station.

what do you think?

>
> Thanks!
> - Tad
>

thx!

-debajit

> -----Original Message-----
> From: Debajit Ghosh [mailto:debajit@google.com]
> Sent: Friday, June 20, 2014 8:57 PM
> To: Tad Coburn
> Cc: Umesh Patil; Christopher Patnoe (patnoe@google.com); Adam Graham
> Subject: Re: CloudQueue API
>
> hey Tad!
>
> things are going well, thanks! hope things are going well with you and the team.
>
> yeah, we're pretty happy with the design and prototyping thus far.
> from your questions, you have a pretty good idea of where we're going

```
> -- the window api and version check api will be similar to what you outlined (with some slightly
different names).  some quick replies:
>
> o cloud queue URLs and REST APIs are going to be pretty similar to what you specified, tho the URLs
won't have the account in them.
> e.g., they'd be something like <host>/cloudqueue/itemswindow and <host>/cloudqueue/version -- the cloud
queue would be implicitly identified by the account (each account would have one) contained in the auth
token.  for future cases where we might need to identify a different account, we'd do so with a
queryparam.
>
> o version ids would be opaque tokens (vs ints).  they could end up being ints, but we'd like to allow
for them to be any kind of opaque token.
>
> o making auth tokens a top-level concept - would it work for you to look for the passed in
Authorization header (in the cloud queue playback request)?  if not, we can certainly make it a special
field.
>
> o fyi, we haven't mapped this to MRP at this time.  the output / design will distill a set of commands
and events which will be sent over the communication channel in our new integration protocol; they should
be mappable as simple extensions to MRP as well.  let's see how we feel about the new discovery and
communication protocol in parallel.
>
> thanks!
>
> -debajit
>
> On Fri, Jun 20, 2014 at 8:57 AM, Tad Coburn <Tad.Coburn@sonos.com> wrote:
>> Debajit -
>>
>>
>>
>> I hope you're week is going well!
>>
>>
>>
>> I want to check in and see if you're happy with your team's progress
>> on the CloudQueue API design.  I'll send a separate email with my
>> ideas on discovering Sonos Groups via SSDP (to keep this email a reasonable length).
>>
>>
>>
>> I heard from Adam/Chris that we might not receive the full CloudQueue
>> API spec from you until mid-July.  I know your team wants to get it
>> really right, but if Sonos waits that long to begin our detailed
>> design and coding we are unlikely to hit the "holiday 2014" deadline
>> for everything on our priority list including the GPM iOS casting to
>> Sonos.  So I want to get started asap J
>>
>>
>>
>> ====CloudQueue API====
>>
>>
>>
>> I'm itching to start prototyping cloud queue playback on our end, and
>> I think we've discussed things enough that I have a good idea of the
>> APIs that your CloudQueue service will expose to our players
>> (receivers).  I think in terms of the player fetching a "window" of
>> tracks from the cloud queue (let me know what term you use).  I'm imagining something like this:
>>
>>
>>
>> fetchTrackWindow(
>>     TIID currentTrackIID,
>>     int numTracksRequested,
>>     int numTracksBeforeCurrent,
>>     OUT uint_32 queueRevisionNumber,
>>     OUT CloudQueueTrack trackWindow[])
>>
>> TIID = Track Item ID
>>
>>
>>
```

```
>> CloudQueueTrack is a structure that contains roughly the following:
>>
>>
>>
>> *         TIID - the Track Item ID
>>
>> *         URI - track URI (equivalent to the "data" URI passed via the MRP
>> API)
>>
>> *         MimeType - MIME type for the track URI
>>
>> *         Metadata fields equivalent to
>> android.support.v7.media.MediaItemMetadata
>>
>>
>>
>> getRevision(
>>
>>     OUT queueRevisionNumber)
>>
>>
>>
>> I'm assuming things get kicked off by the GPM app sending a PLAY
>> action with a CQ REST URI to the Sonos MRP.
>>
>> -> we need a new MIME type for this CQ URI
>>
>> The CQ URI might look like:
>>
>>      http://some.server.google.com/path/cq/CQ1234
>>
>> where "CQ1234" is the ID of a specific cloud queue instance.
>>
>>
>>
>> fetchWindow() could be a REST method that uses a URI like this:
>>
>>
>> http://some.server.google.com/path/cq/CQ1234/tracks?currentTrackIID=<
>> T
>> IID>&numTracksRequested=10&numTracksBeforeCurrent=3
>>
>> which would return a JSON-encoded response containing the OUT
>> parameter data.
>>
>>
>>
>> getRevision() could be  a REST method that uses a URI like this:
>>
>>     http://some.server.google.com/path/cq/CQ1234/revision
>>
>> which would return a JSON-encoded value containing the current
>> revision number for the queue.
>>
>>
>>
>> I think the OAuth token will need to be passed more explicitly to the
>> MRP API, rather than as an opaque "custom HTTP header"; otherwise its
>> not clear how our player will process it and in particular _replace_
>> it when the cloud queue service returns a new value.
>>
>>
>>
>> The Sonos player will pass the OAuth token with all CQ REST calls.
>> We'll use the "Authorization:" header.  The value of that header will
>> be something like "GPlay: <oauthtoken>"
>>
>> -> we need to agree on how the end of the queue is denoted.  Is there
>> -> a
>> special marker (special TIID value), or does fetchWindow() signal the
>> end by returning fewer tracks than were requested?
>>
>>
```

```
>>
>> Is this close?  If so, we can start prototyping basic playback on our
>> end even without your official spec.
>>
>>
>>
>>
>>
>> Best,
>>
>> - Tad
```